STRINGY FLOPPY FOR TIMEX/SINCLAIR by

A&J MICRO DRIVE

1050 E. Duane Ave., Suite I
Sunnyvale, CA 94086
U.S.A.

(408) 732-9292

TIMEX/SINCLAIR EXATRON STRINGY FLOPPY

T A B L E O F C O N T E N T S

I. Introduction

The EXATRON Stringy Floppy (ESF for short) is a mass storage device designed for use with microcomputers. It's probably the most inportant addition you'll ever make to your TIMEX/SINCLAIR computer system; perhaps you'll be surprised to learn what it can do for you!

HERE ARE SOME OF THE OUTSTANDING FEATURES OF THE ESF:

FAST: The EXATRON Stringy Floppy both saves and loads programs and data at a rate almost 30 times faster than cassette recrders -- in fact, it can load a 16K program into your computer in less than 15 seconds!

RELIABLE: The ESF was designed from the ground up to digital standards. All of it is operations are controlled by software, and are highly reliable.

SIMPLE: Convenient, effortless and worry-free operation is provided by the built-in ESOS operating system in ROM. Menus with single-keystroke commands and "plain English" diagnostic messages allow novices to operate the system with ease.

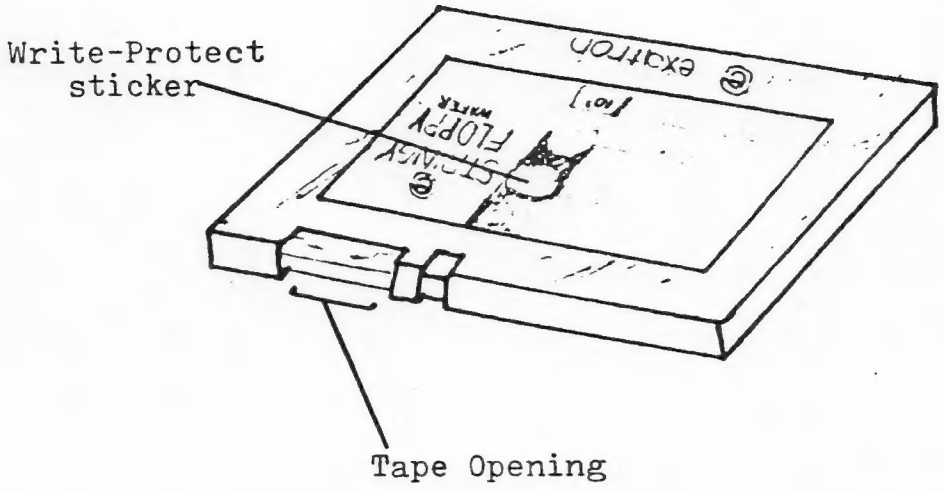
POWERFUL: The ESOS operating system allows program "chaining",editing of previously saved programs, data-only save, and other advanced operations in addition to the maintenence of complete "catalogs" or "directories" for each wafer. You'll never have to guess about a wafer's contents -- ever!

Before beginning setup and checkout of the Stringy Floppy, let's familiarize ourselves with the components of the system as depicted in figure 1:

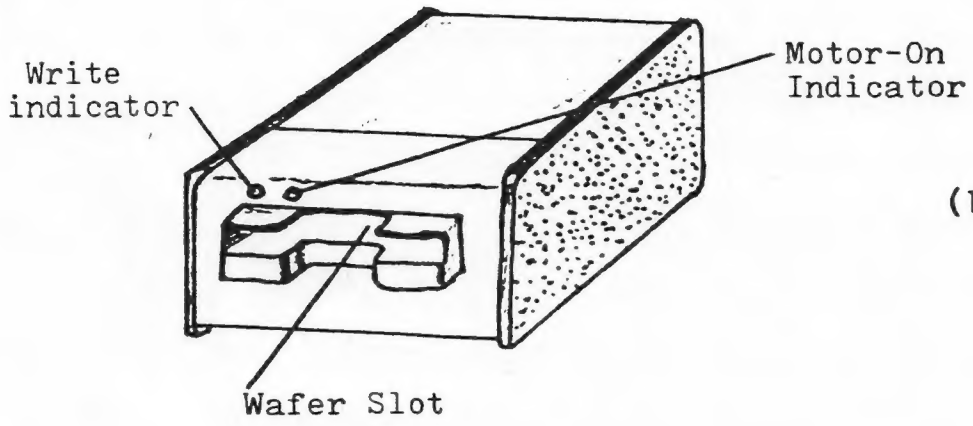
Programs and data are stored on a medium called a WAFER. Inside the wafer is a precision, digital-grade magnetic tape in a "continuous loop" configuration (similar to an 8-track cartridge). A light-reflective splice is placed at the beginning of the tape. The computer can sense the presence of this splice, and thus can tell where each tape begins and ends. (The splice shows throuygh the window on top of the wafer).

Another important feature of the wafer is the "write-protect" sticker. While this sticker is intact on the wafer, programs and data may be saved or "written" to the wafer; when the sticker is covered or removed, the wafer is said to be "write-protected" and may only then be read (loaded) by the computer. This feature is intended to prevent accidental erasure of important material.

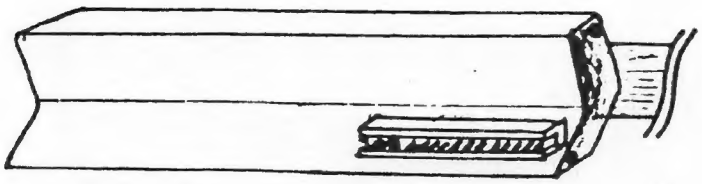
The amount of storage available on a wager is directly related to the length of tape on it. A 5' wafer (the shortest size) cna hold about 7K bytes of material, whil the 50' wafer can hold about 70K!



(a) The Wafer.



(b) Drive Unit



(c) Interface Unit

Page 2 Figure 1. Components of The Stringy Floppy System.

The DRIVE UNIT accepts the wafers and performs operations on them as directed by the INTERFACE UNIT. The drive unit contains the necessary mechanics for moving the tape (motor/capstan, guides, etc) plus the electronics for reading and writing tapes (amplifier). The drive unit also contains special opto-electronic sensors to detect "end of tape" and "write-protect" conditions.

The INTERFACE UNIT can be thought of as an intermediary between the computer and the drive unit. The interface takes commands from the computer and in turn, commands the drive unit to perform the needed operation(s). This action takes place under the scrutiny of a program called ESOS (short for EXATRON Stringy Operating System) which is also contained inside the interface unit. The front of the interface unit has a connector that plugs directly onto the back of the TIMEX/SINCLAIR computer; the back side of the interface has a connector which is an extension of the computer "bus", so that other peripherals may be used along with the ESF.

Some precautions are in order when handling wafers (they are really common sense):

(1) Don't touch the tape or pull out a loop of tape from the wafer. Touching the tape leaves grease deposits, which eventually affect the performance of the wafer-- possibly causing enough of a "drop out" to render a program or set of data useless. Pulling out a loop of tape can lead to binding and/or uneven tape travel, making that wafer unuseable.

(2) Avoid inserting and removing wafers from the drive unit while the motor is running. Otherwise a loop of tape may be pulled out, or worse yet, the tape can become hopelessly tangled!

(3) Keep wafers away from magnets, magnetic fields, electrostatic fields, and X-rays. Information is stored on wafers as patterns of magnetism, and external fields can disrupt these patterns, resulting in loss of data. Strong electrostatic fields can erase material due to " secondary effects." At airports, have luggage containing wafers hand-inspected or remove the wafers before the luggage is to be X-rayed. And of course, avoid laying wafers on any television set!

(4) Electronic equipment is usually unstable during power up and power down, so it is highly recommended that no wafer containing important imformation be left fully inserted in the drive unit at these times; otherwise an electrical transient can cause the drive unit to unexpectedly write a pulse or "glitch" on the wafer, possibly destroying data.

II. Checkout and Installation of The Stringy Floppy

As you unpack your Stringy Floppy, carefully check to see that all components and accessories are present and undamaged. Keep the original carton for re-packing in the unlikely event that repair service is ever needed.

Figure 2a illustrates the various components in a typical TIMEX/SINCLAIR/ESF computer system. The following is the minimum recommended system configuration:

(A) Sinclair ZX80, ZX81, Timex 1000, or 1500.
NOTE: THE ESF WORKS ON A ZX80 ONLY IF IT CONTAINS THE 8K ROM BASIC INTERPRETER.

(B) 16K RAM pack, or equivalent. (The ESF is designed to function with up to 32K RAM!)

(C) EXATRON Stringy Floppy DRIVE UNIT. (An additional drive may be added, if desired).

(D) ESF Interface Unit.

(E) Of course, a suitable television/monitor.

To install the ESF on your computer, do the following:

1. Make sure all power is disconnected from the system; if this precaution is not followed, the computer may be damaged.

2. As shown in figure 2a, install the INTERFACE UNIT onto the back of the computer. To do this, first visually line-up the female connector on the interface unit with the male card-edge of the computer. Note the presence of the aligning "pin" on the interface unit's connector. This pin fits into a notch on the computer edge connector. Once aligned, firmly press computer and interface unit together by pressing directly behind the connector. The fit should be quite firm and the interface should not be loose.

3. Now connect the interface to the DRIVE UNIT. Any of the plugs on the interconnect cable is fine to use, but the farthest plug out is usually the most convenient. Align the "bump" on top of the plug with the notch in the socket on the back of the drive unit, and gently press it in. NOTE: To remove a plug from a drive unit, do not pull on the cable! Instead, press the "ears" on each side of the drive unit connector outward (left and right) and the plug will be ejected from the socket.

4. Connect the 16K RAM pack and/or any other peripherals to the back of the interface unit (this connector is only an extension of the computer bus).

5. Plug the ESF POWER PLUG into an AC outlet (110-135 V. 60 cycle).

NOTE: It's a good idea to use a switched power strip to provide AC power to all parts of the computer system -- it makes for easy and safe turn on and turn off.

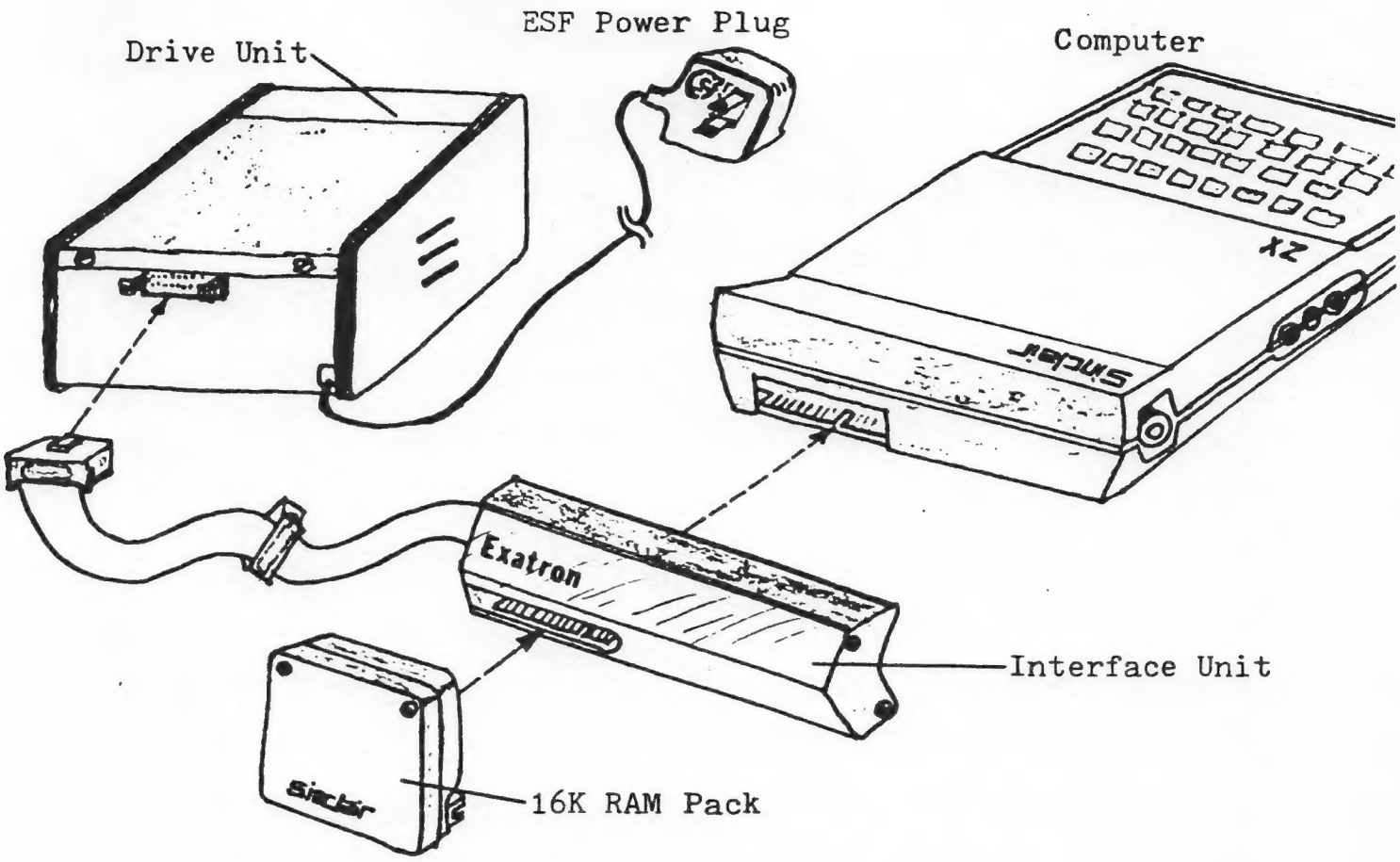


Figure 2 (a) Minimum Recommended System Configuration Hookup.

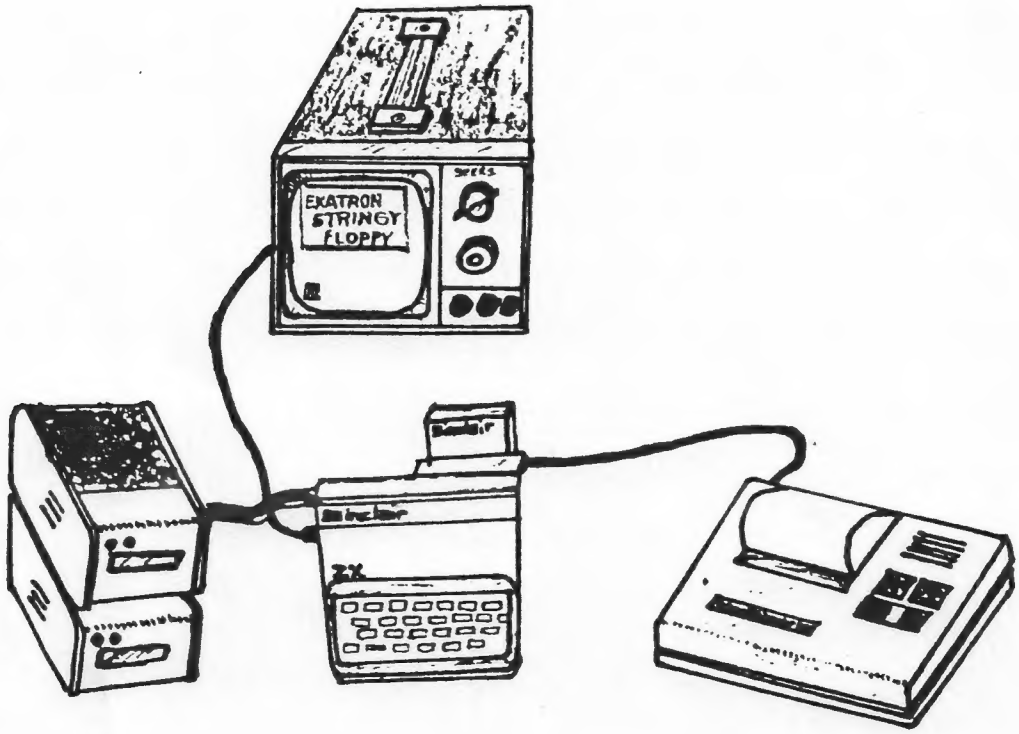


Figure 2 (b) Typical Completed System.

Figure 2b shows a typical complete system including dual ESF drives and a printer.

READY? Power up the system as usual. The "K" cusor should appear on the screen as usual. Right now, the computer doesn't even know a Stringy Floppy exists ...but we're going to change that very soon! Let's get out of BASIC and enter the Stringy Floppy's operating system. Whenever we want to do something with the ESF, we must get into its operating system ... and this is done using the command:

PRINT USR 12345 ""ENTER""

NOTE: We'll use ENTER and NEW LINE interchangeably in this manual. Also, if a word or letter is proceded and followed by double quotes (""ENTER""), it means press this key.

If you are unfamiliar with how to get the "USR" above, here's how:

(1) Press and HOLD the ""SHIFT"" key.
(2) While still holding the shift key, touch ""ENTER"". The cursor should now change to an inverse "F" (for "function").
(3) Release ""SHIFT"" and touch the ""L"" key ...and the word "USR" appears! (whew!)

Enter this command, and the following should appear on your TV screen:

EXATRON STRINGY FLOPPY
OPERATING SYSTEM

1. GET WAFER DIRECTORY
2. LOAD PROGRAM
3. SAVE BASIC PROGRAM
4. INITIALIZE WAFER
5. RETURN TO BASIC
6. LOAD BY NUMBER
7. SAVE MACHINE PROGRAM
8. SELECT DRIVE
9. COPY WAFERS

SELECT?

Take one of the wafers from the kit and place it into the drive unit with the LABEL SIDE UP, and of course, the tape opening facing TOWARDS THE DRIVE UNIT. This is the only way a wafer will fit into the drive. Be sure to push it in until it reaches "home" (final detent). To remove a wafer, simply pull it back out.

Finally, type "4" (No need to press ENTER here), and the computer will take over and proceed to:

(1) Blank out the television screen. The TV picture will always be blank during ESF operations.
(2) Write a special "Test Pattern" over the entire length of the tape. When the motor starts, the RIGHT-HAND L.E.D. on the drive illuminates. When data is being witten to tape, the LEFT=HAND L.E.D. will light up.
(3) Once the tape is written over, the computer will then

verify the quality of the tape by reading back the entire pattern. Then the directory file (explained later on) is written.

(4) Finally, the screen is turned back on, and the "Byte Count" is displayed on the top line of the display. Most informational and diagnostic messages will be printed here. In this case, the message should read:

XXX BYTES FREE

With the rest of the menu below. "XXX" is a number depending upon the length of the wafer.

The entire process can take from 15 seconds to a couple of minutes, depending upon the length of the wafer. If your screen agrees with the above (reads "XXX BYTES FREE"), then you have completed the setup and checkout of your EXATRON Stringy Floppy. You are now ready to proceed to the next section, where we'll explore some of the ins and outs of the ESOS operating system.

IN CASE OF DIFFICULTY...PLEASE CHECK:

DIFFICULTY	PROBABLE CAUSE(S)
Can't get into the menu. Computer just "hangs" or "goes bezerk."	1. DRIVE UNIT not plugged in. 2. DRIVE UNIT not connected to ESF INTERFACE. 3. INTERFACE UNIT not securely attached to computer. 4. The "USR 12345" was mis-typed.
Computer prints "NOT ENOUGH MEMORY" when attempting to get into menu mode.	1. 16K RAM pack (or equivalent) not connected to computer. Use of ESOS requires at least 8K RAM. 2. RAM pack is malfunctioning or poorly seated on back side of INTERFACE UNIT.
Can't even get "K" cursor to come on screen when turning on computer.	1. INTERFACE UNIT improperly seated. 2. RAM pack poorly seated.
After typing "4", an error message appears instead of byte count message: :PARITY ERROR:	1. Loose oxide particles on tape try operation again. 2. Defective wafer.
:WAFER IS WRITE PROTECTED:	1. The white "write protect" sticker is mispositioned or missing.
:BREAK PRESSED:	1. The BREAK or ENTER key was pressed, aborting the operation.

III. Getting Started in ESOS: Saving BASIC programs on Wafers

AS mentioned earlier, each wafer used by ESOS contains a DIRECTORY. This is a special file that stores an ordered list or catalog of programs to be found on the wafer. The directory tells ESOS exactly what to expect from a wafer -- In fact, if a program's name is not in the directory, the ESOS assumes that the program is just not on the wafer!

The only drawback to this system is that wafers do not have directories on them to begin with. Therefore, BEFORE you try to begin saving programs on any wafer with ESOS, you must "Initialize" or "start up" the directory. Once this has been done for any particular wafer, it need not be done again. Here is the sequence for Initializing the Directory, and wafer:

A. Get into ESF's MENU MODE using the command "PRINT USR 12345" as in the previous section.

B. Insert the intended wafer into the drive slot.

C. Type ""4"" to initialize the directory, and wafer. When procedure is finished, the FREE BYTE COUNT will be displayed on top of the screen. This is a continually updated estimate of storage space for programs and data.

D. It's helpful to date wafers when Initializing them, especiallly if you use many wafers per month.

E. To return to BASIC, type ""5"". The "K" cursor will reappear on the bottom line of the display. Note that jumping back and forth betwwen BASIC and MENU mode has no effect on programs or data in the computer.

SAVING BASIC PROGRAMS:

To begin, type in this one-line program:

```
10 PRINT "EXATRON STRINGY FLOPPY"
```

Then get into MENU MODE (via "PRINT USR 12345"), and initialize a wafer if one is not yet initialized; the wafer used in section II will do just fine.

What do you call a program?...Well, in ESOS (as well as in Sinclair BASIC) we call them NAMES! (Sometimes, even the ones that DO WORK get this treatment also!) A more formal term for the "name" of a program is FILESPEC, short for FILE SPECIFICATION. A filespec is nothing more than a "string" of characters made to represent an entire porgram. Under ESOS, up to 8 characters "PROG1" or just about any other symoblic name.

Type ""3"" (to save a BASIC program), and the computer will respond:

```
NAME?#
```

The computer is waiting for you to tell it the name of the resident program(currently in memory). The "#" is the ESOS cursor. Simply type in the desired name and press ""ENTER"". Be sure the wafer is fully inserted in the drive slot when you do this.

After you press ""ENTER"" in response to the "NAME?" query. The computer will proceed to copy the program in the computer out to the wafer; up date the directory to include your program and its name; and finally, verify your program file on tape against computer memory to ensure a good save. All this takes only a few seconds!

NOTES:

(1) When the computer is displaying the "#" cursor, you can do one of the following:
   A. CORRECT MISTAKES by backspacing with SHIFT and 0.
   B. QUIT THE OPERATION by pressing SHIFT and SPACE. This will return you to the menu.

(2) To ABORT an operation while the drive is running, press either the SPACE or ENTER key. The screen will return with message "BREAK PRESSED" to let you know what happened.

(3) Spaces are never allowed inside filenames; hence the computer won't let you type them in!

After completing the save operation, your program is stored on the wafer...but how do you know for sure? Easy. We can look at the wafer directory at any time we're in MENU MODE. Simply type in command ""1"". The drive will start up, load in the directory, and finally the directory will appear on your screen something like this:

```
FILE DIRECTORY-------------ESOS
PROG1
XXX BYTES FREE
NAME?#
```

Looks like it's there, all right -- but why is the computer telling me"XXX BYTES FREE" and ASKING me "NAME?" ??
The message "XXX BYTES FREE" is an estimate of the remaining storage capacity of the wafer.

The "NAME?" question is part of another feature of ESOS: Normally, when you place a wafer into the drive (when starting the system), you'll retrieve or LOAD some program into the computer, although you might not quite be sure of the program's name, or for that matter, what programs are available on the wafer. This feature allows a direct LOAD of a program immediatly after viewing the directory...in short, whatever name you type into the computer will be loaded into the computer. Of course, you can ignore this feature by:
(1) ABORTING by using SHIFT and SPACE.
(2) Pressing ENTER as your sole response. (Here the message "PROGRAM NOT FOUND" will appear).

You can return to BASIC and LIST the program to see that it is still in the computer; it was only copied out to the wafer.

Try to save the program under another name such as "PROG2" Practice getting the directory a few times until you feel comfortable with it; then go on.

A SIZEABLE DEVELOPMENT! ... Let's say you expanded your one-line program to a whopping two-line program!

```
10 PRINT "EXATRON STRINGY FLOPPY"
20 PRINT "MASS STORAGE SYSTEM"
```

Most likely, you want it saved under the old filename, right? Well, go ahead and try it! Get into MENU MODE again and save using command ""3"".

BOMBED OUT, didn't it? The computer should have responded with "FILE TOO LONG" at the top of the screen.

Here's what happened: The computer searched the wafer's directory for the filename you entered (PROG1). Since it already had a "PROG1" file, it then checked the LENGTH of the existing PROG1 file against the length of the program currently in the computer. It would have let you save the program, except that the requested file length was larger than the original file length allocated -- a definte "no-no!" The computer stopped, and gave you the error message.

However, there is a neat way of saving expanded programs like this one. The ideal is to somehow "reserve" a longer file than is actually needed for the FIRST save of the program. Then all future versions will fit into this file. But how do we get a longer file to start with?... Answer, have a longer PROGRAM to start with!

We can "fool" the system into THINKING that there is a long program in the computer when in reality only a one-liner exists. Actually, we won't be "fooling" the system at all. We'll just take advantage of the fact that VARIABLES as well as PROGRAM TEXT are preserved during program saves. VARIABLES includes any and ALL BASIC VARIABLES: Scalars, Arrays; numeric and strings. ALL ARE PRESERVED, and ALL ADD TO THE FILE LENGHT.

So, before that first save, we can reserve extra file lenght by simply defining extra variables. The easiest variable type for using up memory space is the ARRAY. Perhaps an example is in order: Suppose the 2-line program above is in the computer, and we want to reserve an extra 500 bytes of storage in its file for expansion.

#Bytes reserve

STEP 1. Pick an UNUSED ARRAY NAME and DIMension it to: 5
So: DIM A(100) reserves approximately 500 bytes.
(as a direct command in BASIC) would do the job.

STEP2. Save the Program by going into MENU MODE. In our case, we'll have to name it something other than PROG1. Let's call it LARGE1.

STEP3. The next time the program is to be saved, execute a CLEAR command before jumping to the menu. Then the program will thereafter be re-saveable (at least until its length exceeds the limit!) The purpose of the CLEAR command is to erase the array (Variables will be lost at this time, saves will fit into the existing file.

NOTE: On the first save, a program need not be in the computer at all! Dimensioning an array at this time allocates total file length.

IV. Loading Programs Into The Computer

Loading program material back into the computer from wafers is a very simple operation. There are actually three distinct methods that can be used (the first two are covered in this section).

1.) Using the program's FILENAME.
2.) Using the program's "PHYSICAL RECORD NUMBER"
3.) From inside BASIC using the PRN and a "USR" CALL to the ESF firmware.

To load a program when the filename is already known, simply get into MENU MODE (via USR 12345) and type ""2"". The computer will respond with:

NAME?#

Then you can type in the name of the program you want, pressing ""ENTER"" when done.

If you don't know the exact name of the program and wish to see the directory, type ""1"" and the dirctory will be displayed, and the computer then will request the NAME.

A message you might see is "PROGRAM NOT FOUND". If any mispelling of the filename occurs, the computer cannot make a proper "match" so use care when entering filenames.

- USING THE PHYSICAL RECORD NUMBER (PRN)

Sometimes it's possible to save a little time by supplying ESOS with a number instead of a name to locate a given program. This is possible because all files and programs on a wafer are stored by number in the first place; only that fact is normally transparent to the user since ESOS manages the numbers and filename internally. Figure 3 illustrates the file structure of an ESOS wafer. Notice that file 1 is ALWAYS the directory.

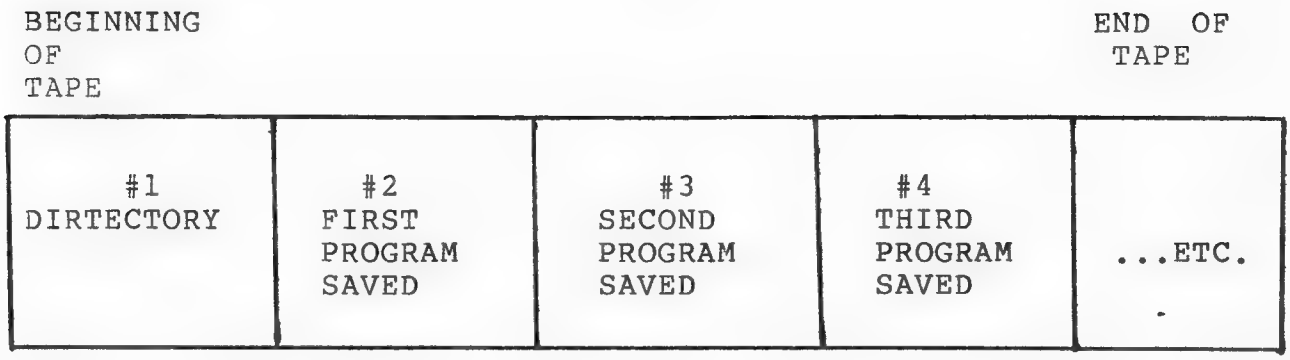
Files 2 and on are program files. Programs are stored in these files in simple numerical order. For example, the FIRST program saved on a wafer goes in file 2, the SECOND goes in file 3, the THIRD in file 4, and so on.

You may have already noticed, but the order of files as stored on the tape is the same order as they are printed out in the directory -- so it's quite easy to determine where a program is stored. Suppose a directory looked like this:

```
WAFER DIRECTORY ---------------ESOS
HERS
HIS
PLOTTER
21232 BYTES FREE
.
.
.
```

What PRNs correspond to each program file? Simple. Just take the TOP file (HERS), call it 2, and go up from there. So:

HERS = 2 , HIS = 3 , PLOTTER = 4.



File Organization. Note how the DIRECTORY is always the first file. The broad lines separating files are symbolic of file marks. A tape may contain up to 127 different files.

The idea behind using PRNs is this: Using a PRN instead of a filename can be faster, since an extra pass of the tape can often be avoided if the computer does not have to search out the directory. Instead, you are telling it "where to go!"

Command ""6"" from MENU MODE allows the computer to accept a file number. Typing this will result in:

ESF FILE NUMBER?#

On the screen. Type in the PRN (for example, to load in PLOTTER, "4" would be used), and press ENTER.

PRECAUTIONS:

(1.) Never use a number less than 2 for a PRN, or the directory file may be loaded in as a BASIC program. This will ALWAYS crash the system.

(2.) If you enter a number that is too high (higher than the last file on your tape), the Stringy Floppy will never stop since it won't be able to locate the file. If this should happen, touch ""BREAK"" to regain control.

- LOADING OF MACHINE LANGUAGE PROGRAMS:

Machine language programs come in two different varieties: First, there are many that are really part of a BASIC program as loaded from cassette in Sinclair BASIC. The ESF treats these as BASIC programs (e.g., you save the "BASIC" program and the embedded assembly-language code is saved also).

The second variety of machine language programs are those saved directly from memory using the ESF and/or ESF monitor program (optional/extra). The procedure for loading BOTH varieties of machine language programs is exactly the same as loading a BASIC program, with following exception:

* A MACHINE-LANGUAGE PROGRAM SAVED DIRECTLY FROM THE ESF OR ESF MONITOR (OPTIONAL/EXTRA) IS IN A DIFFERENT FILE FORMAT THAN A BASIC PROGRAM. ESOS RECOGNIZES THIS SPECIAL FORMAT, AND TRAÑSFERS CONTROL TO THESE PROGRAMS IMMEDIATLY AFTER LOADING IS COMPLETE. THIS PROCEDURE IS OFTEN REFERRED TO AS AN "AUTOSTART" AND IS EXPLAINED LATER ON. *

V. Chaining Programs

Using some of the ideas from the previous section, it's possible to "chain" programs in BASIC. To "Chain" porgrams means to have one program load in another, and give control of the system to it. In effect, the new program should RUN once it has been loaded into the computer. The ESF firmware does precisely this.

Before the firmware can chain programs, it needs to know two things: First, it needs a "command" so it knows what YOU want it to do; and second, it needs a direct object or "parameter" so that it knows what to do it to. Well, Sinclair BASIC has no direct way of doing this, but we can get very sneaky and enter via the back door. Refer to figure 4 and consider the following:

A. We'll pick two memory locations, one for command and the other for parameters. The ESF firmware already "knows" where these memory locations are.
B. Next, we'll place numeric codes in these locations that tell the computer (ESF) what to do.
C. Finally, we'll give the ESF FIRMWARE control of the situation: It will be able to look at these locations and tell what we want done. The firmware will do the operation and return control to us (inside the BASIC program).

The command location is 16451, and the parameter location is 16390. Initialize a wafer and enter the two programs below on it IN ORDER, and RUN either one of them. Something very interesting will happen:

PROGRAM 1:

```
10 CLS
20 PRINT"THIS IS THE FIRST PROGRAM."
30 PAUSE 100
40 POKE 16451,1
50 POKE 16390,3
60 LET X=USR 12291
```

PROGRAM 2:

```
10 CLS
20 PRINT"THIS IS PROGRAM 2"
30 PAUSE 100
40 POKE 16451,1
50 POKE 16390,2
60 LET X=USR 12291
```

The action statements in each program are lines 40, 50, and 60. One program will call the other, which calls the original again -- an endless loop! As shown in figure 4, line 40 POKEs the value 1 into memory location 16451: To the ESF firmware, this means"Chain BASIC programs".

Line 50 POKEs the desired PRN into location 16390. Thus the firmware can tell what file number to load in and execute. (If you are unfamiliar with PRN, refer to section IV). The "action" occurs in line 60, where a USR CALL is made to the portion of the ESF firmware beginning at 12291.

ESF COMMAND CODES

CODE	MEANING
1	Load and RUN program file ("CHAIN")
3	CERTIFY the Wafer
5	Select additional ESF Drive Unit
6	Write all Variables to a Data File
8	Read in all Variables from a Data File

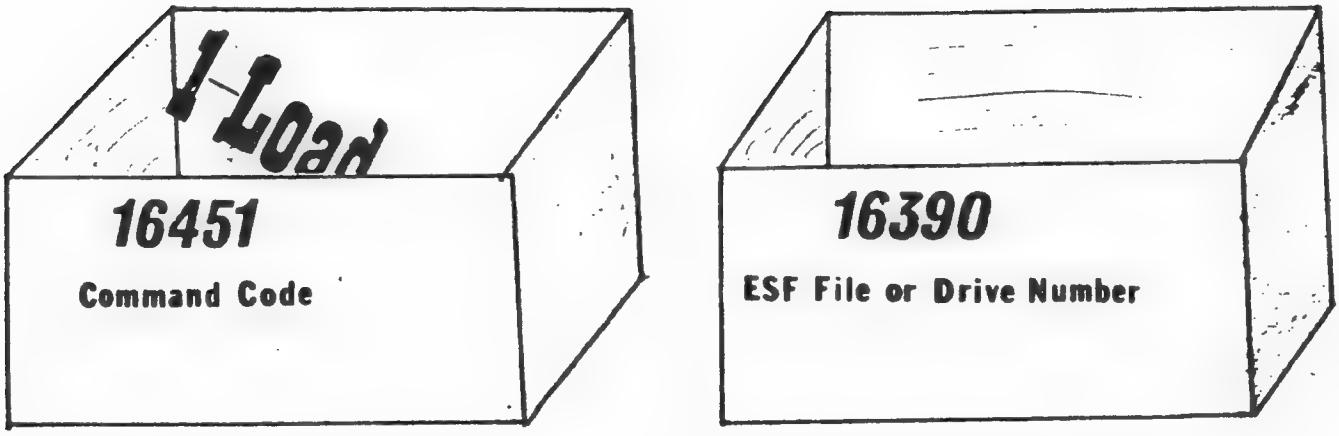


Figure 4: Special Memory Locations used by ESF

Notes about program CHAINING:

1. The USR CALL is ALWAYS made to 12291.

2. When the new program is loaded, its variables are loaded as well; to preserve the current variables, save them in a data file (Described in section VI Other Operations in BASIC).

3. The POKE statements should always immediately precede the USR CALL statement, or unreliable operation may result.

4. If any errors occur during ESF Operations with BASIC, the message appears on top of the screen, and control is returned to basic with a "F" or "G" cursor.

VI. Other Operations in Sinclair BASIC

Figure 4 shows some other operations that can be completed while in a BASIC program, but were not discussed in section V. These operations are:

(1) Certification of Wafers
(2) Selection of additional ESF Drive Units
(3) Data I/O (Storage & Retrieval of all Variables)

- CERTIFICATION OF WAFERS:

For Data I/O applications (and possibly others in the future), it might be necessary to erase and certify the quality of a wafer from BASIC. The wafer is left empty and ready to receive files starting at file #1 (NO DIRECTORY IS WRITTEN). A typical sequence for certifying a wafer is as follows:

```
1000 PRINT "STAND BY,CERTIFYING"
1010 PAUSE 100
1020 POKE 16390,0
1030 POKE 16451,3
1040 LET X = USR 12291
1050 PRINT "OPERATION COMPLETED."
```

CAUTION: If you try this example, don't leave a wafer with any important material on it in the drive slot; use a blank or unimportant wafer, as all contents will be erased!

NOTE: Before using any CALLs to the ESF firmware via the USR function, the system must be in and out of MENU MODE at least once (so that the hardware will be properly initialized). This is not normally a problem, since the first program is usually loaded from the ESF via menu mode upon system power up. As long as power is maintained to the system, the hardware will remain initialized.

- SELECTION OF ADDITIONAL ESF DRIVE UNITS

It's usually convenient to use an additional drive for data I/O purpose (although data files can be on the same wafer as programs). The DRIVE SELECT function switches in additional drives for you. The following program will select drive #2, prompt the operator, and certify the wafer in drive #2:

```
900 POKE 16390,2
910 REM SELECT DRIVE 2
920 REM
930 POKE 16451,5
940 REM COMMAND CODE 5=SELECT DRIVE
950 LET X = USR 12291
960 PRINT "PLACE BLANK WAFER IN DRIVE 2"
970 INPUT A$
980 GOTO 1000
990 REM GOTO"CERTIFY WAFER" ABOVE
```

It's possible to save the variable area onto a wafer using ESOS from BASIC. Individual variables are NOT saved one at a time. This method is both simple and efficient, and is directly suited for applications such as data bases, file editors, and more.

When the variable area is retreived from wafer, all the variables previously stored become "active" or "real" again, including any variables used in FOR/NEXT loops. This requires some special handling, but is not too tricky to accomplish.

Let's create two programs. One will ask the user for a line of text, then store the text in a data file. The second program will retreive the text and print it. If we start with a freshly-initialized wafer, the programs will be stored in file numbers 2 and 3. Therefore, if nothing else is to be placed on the wafer by ESOS (programs), it's perfectly okay to use file 4 for the data file!

PROGRAM 1:

```
10 PRINT "ENTER A LINE OF TEXT?"
20 INPUT A$
30 PRINT A$
40 LET SERIAL=INT(RND*5000)
50 REM WHY NOT HAVE A
60 REM "SERIAL NUMBER"??
70 REM NOW SAVE DATA FILE
80 POKE 16451,6
90 POKE 16390,4
100 LET X = USR 12291
110 REM NOW GO TO PROGRAM 2
120 POKE 16451,1
130 POKE 16390,3
140 LET X = USR 12291
```

PROGRAM 2:

```
10 CLEAR
20 REM CLEAR MUST ALWAYS BE USED
30 REM BEFORE DEMENSIONING VAR
40 REM TABLE SPACE.
50 DIM A(20)
60 REM RESERVE 100+ BYTES IN
70 REM VAR TABLE.
80 POKE 16390,4
90 POKE 16451,8
100 REM PREPARE TO READ VARIABLES
110 LET X= USR 12291
120 PRINT"THE LINE YOU TYPED:"
130 PRINT A$
140 PRINT
150 PRINT"SERIAL NUMBER";SERIAL
160 STOP
```

Program 1 inputs the data (A$) and stores it, along with a "serial number" (for fun) in a data file. The program chaining feature is then used to turn control over to PROGRAM 2.

Program 2 immediately loads in a data file (located as record 4) and prints the result. Command code 8 is used to read back the data (as in figure 4). However, before the data can actually be loaded into the computer, two things must be done:

1. Execute a CLEAR statement to erase all current variables.

2. Dimension a numeric array to 5 times the number of bytes needed to load in the variables. (you can go over this amount, and ESOS will compact the loaded file as necessary to conserve memory). If you do not reserve enough space, ESOS will return the message: NOT ENOUGH MEMORY. Program 2 reserves a little more than 100 bytes, since 20*5=100.

NOTE: After step 2, don't create any new variables, or the system can become confused (operation of a loaded data file can become erratic).

Remember that once a data file is loaded, the variables in use when the file was stored are all restored - including loop variables.

VII. Using COPY to make Backup Copies of Wafers

The Stringy Floppy has a built-in command for copying wafers. It provides a fast and sure duplication. However two ESF drives are required to use this command: The contents of the wafer in drive unit 1 are copied out to the wafer in drive unit 2.

To make a copy, proceed as follows:

First, take a blank wafer of the same length as the donor wafer and initialize it in either drive (using the DRIVE SELECT command if necessary). Using a wafer of the same length as the donor ensures that the "byte count" will be approximately correct.

Next, place the blank in drive 2, and the donor in drive 1. Type ""9"" and the computer will respond with:

READY DRIVES?#

Press ""ENTER"" and the copying will begin. The system first seeks the directory of drive 1, then copies the directory out file 1, drive 2. Next, file 2 of drive 1 is read in, and written out to drive 2, file 2. The process continues until all files have been copied.

NOTE: If a freshly initialized wafer is placed in drive 1 prior to the copy command, the system aborts with no operation attempted (This feature protects you in the event that the donor and recipient wafers accidentally get reversed).

If you attempt to make backup copies when drive 2 is not connected or malfunctioning, the message:

DRIVE NOT PRESENT

Will warn you of the problem.

IMPORTANT: Files written to drive 2 are not verified by the system. Although the ESF is extremely reliable, you should go back and double check the recipient wafer after a backup if the application is critical.

VIII. Saving Machine-Language Programs From The Menu

While an understanding of machine language is not necessary for the operation of the TIMEX/SINCLAIR ESF, one may on occasion wish to save a certain machine language program to wafer. Command ""7"" from the MENU MODE performs this function. This function will save any block of memory; and there need not be any program in the block at all, since the ESF simply copies the block out to wafer without making any test of its contents.

Upon entering command ""7"", the following dialogue takes place:

START ADDRESS?# (Computer wants to know first address in memory block to be saved).

LENGTH?# (Computer wants to know HOW LONG the recorded file is to be. The last address saved is equal to the START PLUS the LENGTH, minus 1).

AUTOSTART?# (When a machine language programs is loaded, the computer automatically executes it. It jumps to the "autostart" address. AUTOSTART ADDRESS is part of a machine language file.)

NAME?# Assigns a filename to the program.

NOTE: With a long sequence of inputs like this one, it's possible to run out of room on the screens. The screens will clear and proceed with the next question in this case, so don't be alarmed.

The procedure for loading a machine language program is the same as that for loading a BASIC program (Section IV), except that execution begins automatically at the AUTOSTART ADDRESS with machine language programs.

A VERIFY ERROR will occur if the saved program does not verify with the version in memory. Due to the design of the TIMEX/SINCLAIR hardware, certain areas in memory can change periodically. Areas to avoid include the top 16K of the memory plane (used for video generation) and any unused blocks of memory (e.g., blocks where no device exists).

Take a fresh wafer and ceritfy it using command ""4"". The computer will respond with:

ESF FILE NUMBER?#

Enter ""1""(to erase all files beginning at file 1, and above) and press ""ENTER", of course. The tape will then be certified by the computer (totally erased, and quality verified).

Now you're ready to save the program to the wafer, so type ""1"" to do so. The computer will again respond with "ESF FILE NUMBER?", and you'll have to tell it ""1"" since this the FIRST program saved on the wafer. The ESF will then proceed to save your program on the wafer.

A second program stored on the same wafer would be given the file number 2; a third, 3; and so on. They are assigned in counting order.

To load in a program saved in ECONOMY MODE, use command ""2"" and supply the file number of the program you desire. If the program is less than 15K, the "LOAD BY NUMBER" command of MENU MODE can load it also.

To make a return to Sinclair BASIC from ECONOMY MODE, use command ""3"".

REMEMBER: You don't need to use ECONOMY MODE unless a 16K program needs to be saved when only 16K RAM is available.

NOTE: On the 1K and 2K RAM systems, ECONOMY MODE will work, but its benefits will not be fully utilized.

IX. ESOS Memory Requirements, and Economy Mode

The ESOS operating system in your TIMEX/SINCLAIR ESF provides a great many additional functions, but there is a "cost" for all of this, and you should be aware of it. To provide for directory and file manipulation, ESOS grabs the top 1K of RAM memory away from the system once MENU MODE has been entered. Thereafter the memory capacity of the system is 15K. If you have a 32K RAM pack installed, a bonus awaits you: ESOS always hunts for the "top" of memory whenever MENU MODE is entered, and resets the "END OF RAM" pointers for Sinclair BASIC to a position that is 1K below the "top." Thus, 32K RAM users will have 31K memory present -- and no more pokes to worry about to get it! Simply enter MENU MODE and the job is done.

What if you have a 16K program and desire to save it on the Stringy Floppy? No problem...we've got you covered here. The solutions are as follows:

(1) Purchase a 32K RAM pack; then you'll have 31K RAM "to burn!"
(2) Use a different mode of the ESF known as ECONOMY MODE.

ECONOMY MODE is really a compromise of sorts. You get the most basic features of the Stringy Floppy, without any filehandling whatsoever. YOU decided what filenumber must be used for the program(since there is no "directory" in ECONOMY MODE either!!) You must also keep track of what is stored on wafers written in ECONOMY MODE.

Remember figure 3? This was a simplified diagram of an ESOS-created wafer. Wafers created in ECONOMY MODE look like this too, with one exception: THERE IS NO DIRECTORY, SO PROGRAMS START IN FILE 1.

To use ECONOMY MODE, do the following:

(1) Power up the computer (or do a PRINT USR 0) to ensure that the memory pointers are at the top of memory. IF YOU EVER ENTER MENU MODE. THE POINTERS ARE PUSHED DOWNWARD AND ARE NOT RESTORED IN ECONOMY MODE!

(2) Load in the 16K program from cassette, or type it in, should you feel especially lucky. Keep a rabbit's foot near the computer in either case...

(3) Do a PRINT USR 12348 command. The screen should show:

```
ECONOMY MODE

1. SAVE PROGRAM
2. LOAD PROGRAM
3. RETURN TO BASIC
4. CERTIFY

SELECT?
```

	EXPLANATION
DRIVE NOT PRESENT	You have attempted to access an ESF Drive Unit which does not exist.
BREAK PRESSED	The BREAK-SPACE or ENTER key was pressed during an ESF operation.
NOT A BASIC PROGRAM	An attempt was made to overlay an ESOS file with a BASIC program file, when the file descriptor indicated that the file-type was not BASIC, and therefore wrong.
PROGRAM NOT FOUND	ESOS could not identify the filename you gave it.
FILE TOO LONG	The resident BASIC program is longer than the file it was intended to be stored in.
VERIFY BAD	The tape and memory versions of a program do not agree during save.
WAFER IS WRITE PROTECTED	The Write-Protect sticker is missing from the wafer (see figure 1).
OUT OF TAPE	There is not enough room left on the tape to store your program or data.
PARITY ERROR	A bit was lost during load or certify.
INCORRECT CHECKSUM	The calculated checksum does not agree with the checksum written on the tape during a load operation.
NOT ENOUGH MEMORY	Insufficient RAM left to accomplish requested rask (usually loading a program or data set). During data I/O, this indicates that the reserve size is too small.
EOF MARK DETECTED	Something distorted or destroyed a file on the tape, allowing the system to skip to the next file mark.

XI. Assembly Language Operations

IMPORTANT: This section is intended for experienced assembly language programmers only, and you can ignore it if you like. No assembly or machine language knowledge is necessary to operate the ESF.

The 4K Firmware contains many useful subroutines for assembly programmmers. Most return with the Z flag set or reset to indicate success or failure. When the Z flag is reset (Z=0), an error code may be in the A register as follows:

CODE	MEANING
01H	WRITE-PROTECT ERROR
02H	BREAK/ENTER PRESSED
04H	TAPE TOO SHORT
08H	PARITY ERROR
10H	CHECKSUM ERROR
20H	OUT OF MEMORY
40H	VERIFY ERROR
80H	EOF MARK DETECTED

IMPORTANT: TO USE TAPE I/O ROUTINES, "FAST" MODE MUST BE SET!

LOCATION	COMMENTS
3000H	Reads in an ESF file. Before calling, set HL=buffer start, and BC to maximum length. Upon return, BC=actual length. In addition if C=0, a machine-language program file has been read; so HL=start, BC=length, and DE=autostart. Tape must be positioned to beginning of file with a call to 300FH BEFORE calling 3006H. Z=1, successful; else error code in A.
3006H	Writes a machine-language file. HL=start, BC=length, DE=autostart. Before calling 3009H, position tape to "beginning of file" with a call to 300FH.
3009H	Writes normal ESF file. HL=buffer start, BC=length. Before calling position tape with a call to 300FH.
300CH	Locates start of a file. File number is in A-register prior to call.
300FH	Writes EOF mark record. A=file number prior to call.
3894H	Prints unsigned integer in HL register at current cursor location. ALL registers are used.

LOCATION	COMMENTS
3015H	Use to input strings of characters from the keyboard. This routine prints a "prompting string" that is pointed to by the HL register pair, and terminated with an FFH byte. A "?" mark is then printed, and input with a "#" cursor commences. The input line is not stored at (ELINE) but instead is buffered at 4047H (part of the line print buffer). Spaces are not allowed; buffer is terminated with a hex 00. If C not = 0, a SHIFT/SPACE sequence was initiated. Up to 9 characters are allowed in the buffer.
3018H	Clears screens, and resets ESF's line-count to 00.
301BH	Prints individual characters placed in the A-register. "Odd" character codes will not cause the routine to crash; if the screen becomes too full, it clears itself.

Other machine language operations are possible with the ESF, and will be covered in a separate, comprehensive manual called "The ZX/TS ESF Assembly Programmer's Guide."

12:000 > ROM 32K
16:000

MEMOTEXT

TURN OFF

A = TOP OF CONN
B = BOTTOM
(SIG NAME)

A & S STRINGY FLOPPY INTERFACE

A0 = 1
A1 = 2
A2 = 4
A3 = 8
A4 = 16
A5 = 32
A6 = 64
A7 = 128
A8 = 256
A9 = 512
A10 = 1024
A11 = 2048
A12 = 4096
A13 = 8192
A14 = 16384
A15 = 32768

2732 ROM SELECT = $\overline{A15} + \overline{A14} + A13 + A12 + \overline{MREQ} + IOREQ + RFSH$

8192 + 4096 = 12288 ; TRUE = 0 ; NOT TRUE = 1

5V

2732 pinout:
B14 (A12) 1 — 24 C
B19 (A7) 1 — 23 (A8) B18
B20 (A6) 2 — 22 (A9) B17
B21 (A5) 3 — 21 (A11) B15
B22 (A4) 4 — 20 (RD) A16
B10 (A3) 5 — 19 (A10) B16
B9 (A2) 6 — 18 C
B6 (A1) 7 — 17 (D7) A1
B7 (A0) 8 — 16 (D6) A7
A4 (D0) 9 — 15 (D5) A8
A5 (D1) 10 — 14 (D4) A10
A6 (D2) 11 — 13 (D3) A9
GND 12

LOW ENABLES

A15 (IOREQ) A23 (RFSH) B14 (A12) B13 (A13)

2.2K 2N3904

ROM CS B23

B11 (A15) B12 (A14) A14 (MREQ)

(A4) (A5) B21 (A6) B20 (A7) B19 (A8) B18 (A3) B10 (A9) B17 (A10) B16

TO BUS 74LS138

WR Y5 RD Y2

74LS125

A9 (D3) (D0) A4 (D2) A6 (D7) A1

CLEAR

SPARE

Y7 = 0 0 1
Y6 = 0 1 0
Y5 = 1 0 1
Y3 = 1 1 0
Y0 = 0 0 0

Y4

2.2K +V 10MFD 35V

74LS138
PARTIAL ENABLE

A10	A9	A8	A7	A6	A5	A4	$\overline{A3}$
1	5	2	1	6	3	1	NOT
0	1	5	2	4	2	6	8
2	2	6	8				
4							

AND OR
A2 A1 A0

2032 TO 2039

A & J FLOPPY INTERFACE